NYU Tandon School of Engineering
CS-GY 6083, Principles of Database Systems, Fall 2024
Prof Phyllis Frankl

Practice Material #3

Problem 1:

In this problem, you have to write SQL and RA queries for a database modeling a subway management system similar to the MTA, given by the following relational schema:

Passenger (pid, pname);

Card (cid, pid, cexpireday, cbalance);

TimePrice (mday, mprice);

Station (sid, sname, saddress);

AddTime (cid, attime, mday, moneypaid);

AddValue (cid, avtime, money);

Ride (cid, sid, rtime, rprice);

In the schema, we have passengers identified by a pid, and cards identified by a cid. There are two alternatives for passengers to pay for rides. The first one is to add time to their cards, so that they can take the train for unlimited times during the time-period. The TimePrice table stores two time periods, 7 days and 30 days, and their prices. The other alternative is to add value to their cards. The value will only become available when any available time period on the card expires. When passengers add time or value, the database stores the number of days or amount of value they add, and the time of the transaction on the card. Also, whenever a passenger enters a station, the system records the time and, if the card that is used has no remaining time but has value, the price is deducted from the current balance in the Card table. As for the MTA system, passengers only swipe when they enter the subway, not when they leave. However, differently from the MTA system, we assume that, when extra time is added, the time period starts immediately, while in the MTA system the time period starts when the card is used the next time.

(a) Draw an ER diagram that is consistent with the above schema. Identify any weak entities.

(b) Identify suitable foreign-key relationships for the above tables.

(c) How would you change the schema if a card can be jointly owned by several passengers?

(d) How would you change the schema if the price is determined by both the departure station and arrival station, instead of one price per ride? Discuss briefly.

(e) Create the above original schema (meaning, not the one from parts(c) and (d)) in a database system, choose appropriate attributes types, and define primary keys, foreign keys, and other constraints. Data for this schema will be made available on NYU Classes; please use that data. Load the data into the database using either insert statements or the bulk-loading facilities. You may use MySQL, Oracle, or

Postgres. If you wish to use a different relational database management system, check with the TAs first.

(f) Write the following SQL queries and execute them in your database system. **Show the queries and the results**:

(i) For each passenger, output how many times they took the train on Dec. 25th, 2019.

(ii) For each passenger, output the total amount of money they spent in 2018 on refills (add value and add time).

(iii) Output the average number of cards each person has.

(iv) Output all passengers who have taken the train more than ten times in a single day.

(v) Output the name of the most popular station (i.e., the station where the most people entered) in 2018.

(vi) Output the cids of any cards that added 7 extra days, but then were never used during those 7 days.

(g) Write SQL statements to perform the following updates to the database:

(i) For all cards that still have at least 10 days of unlimited rides left, extend their time period for one day.

(ii) Delete all the cards with no balance and no remaining time.

(iii) Imagine that a passenger, who is already in the database system, adds 30 days of unlimited rides. Assume also that the card already exists in the database. Write queries to modify the database appropriately, by modifying the Card and AddTime tables.

Note: You may choose the passenger and the card, and directly use the pid and cid instead of finding them among the schema.

Problem 2:

In this problem, you have to create views, write queries that use the views and create triggers for the schema in Problem 1. Execute everything on your database system and report the results.

(a) Define a view that shows only passengers that never had an unlimited ride card during 2018 (i.e., they always used value instead of time), along with the history of their rides. The view should have attributes pid, cid, ttime, and tprice.

Note: a person could have several cards.

(i) Using this view, output the pid of any person who would have saved money by buying a 30-day pass for June 2018.

(ii) Write any other interesting query of your choice that uses this new view

(b) Write a trigger that automatically reduces the balance on a card when it is used to enter a station, if the card has no remaining time left and there is enough value left on the card.

(c) Write a trigger that automatically adds an extra 5% bonus to a card whenever at least 5 dollars of value are added. Thus, a $20 purchase gives a person $21.00 extra credit on the card.

Problem 3:

In this problem, you need to design a relational database for a meal service that provides people with a choice of lunch sets. For example, imagine a restaurant or food truck that offers lunch sets for workers in a factory or in an office park. Customers interested in the service must sign up with their name, contact info, and credit card. Before the beginning of each month, customers buy a certain number of “lunch credits” according to a price plan given by the service; e.g., a customer may buy 100 credits for $100, 150 credits for $140, or 200 credits for $180. The service offers several different meals, where each meal has a name, and a longer description, and maybe additional nutritional information. For example, there might be a meal *“Fettucine Alfredo: Fettucine in our own Alfredo sauce with a side of broccoli. 500 calories. Vegetarian”*.

However, on every given day, the service only offers a small selection of meals from the overall menu. For example, on a particular day, the service may offer a choice between three options, *“Fettucine Alfredo”* for 9 credits, *“General Tso’s Chicken”* for 8 credits, *and “Caesar Salad”* for 6 credits. The prices of the meals may vary between different days, so you should not assume that *“Ceasar Salad”* is always 6 credits.

The available meals from the menu (called the “daily choices”) change from day to day, and will be sent to the customers a few days in advance, so that they can select their option -- or choose not to have lunch that day. This way the service can plan ahead and knows how many meals to prepare. However, customers have to spend their credits in a given month or they lose them at the end of the month, so this is a service for people who regularly order from the service, with discounts for people buying more credits.
Note: The service may also offer an option to roll over a limited number of credits, say up to 30, to the next month, or may allow people to purchase extra credits when they run out, but you do not have to model these options in this homework.

Your task is to design a database that keeps track of customers and their credits and meal choices. You should also keep track of the meals that are on the menu, and store which meals were offered to customers on any given day. You do not have to worry about how the meals reach the customer, whether they are delivered or picked up by the customer.

(a) Design an ER diagram that models the above scenario. Identify suitable keys and the cardinalities of the relationships. Also identify any weak entities. Discuss any assumptions that you are making in your design.

(b) Convert your ER diagram into a relational schema. Show primary keys and foreign key constraints.

(c) Write statements in SQL for the following queries. Note that, if your schema does not allow you to answer a query, you may have to go back and change your design.

(i) For each meal on the menu, output on how many days it was offered as a daily choice in 2018, and how many times it was picked by customers (meaning, how many portions were sold).

(ii) Output the customer ID and name of any customer who lost more than 20% of their paid credits during 2018, due to credits expiring at the end of a month.

(iii) We say that a menu item A is beaten by another item B if item A is significantly less popular than item B when both A and B are offered as options on the same day. In particular, we require that on those days where both A and B were offered in 2018, overall there were at least twice as many orders of B than of A. Output all pairs of items A and B where A is beaten by B.

(d) Create the tables in your database system, and insert some sample data (maybe 5-10 tuples per table). Choose an interesting and meaningful data set. Then execute the queries from part (c). Submit your sample data, and screenshots or logs of the executed queries and the relative outputs.

Problem 4:

For the given schema below:

Student (<u>sid</u>, sname, semail, sphone);

Club (<u>cid</u>, cname, cdescription);

Event (<u>eid</u>, ename, edescription, edate, memprice, nonmemprice, maxpeople)

Membership (<u>sid, cid, semester, year</u>, memberfee);

sid references Student(sid)

cid references Club(cid)

HoldsEvent (<u>eid, cid</u>);

eid references Event(eid)

cid references Club(cid)

Register(<u>eid, sid</u>, price, rating);

eid references Event(eid)

sid references Student(sid)

Assumption: We assume all students attend the events they registered for. Each year has only two semesters, Fall and Spring. Besides, you can assume there is a **function used to map dates to semesters** for this problem or you can assume **09-01 to 01-01 will be Fall semester and 03-01 to 06-01 will be Spring semester**.

Write queries for the following tasks:

1. For each student, output the number of events the student registered for in Fall 2023. (Hint: As there may be students attending 0 events, a natural join is not enough.
2. Output the name of the club that has the greatest increase in member fees in Spring 2024 compared with Fall 2023.
3. Output the id, name, and date of the event co-organized by the largest number of clubs.
4. Output the id and name of the events held in Fall 2023 that have a number of attendees equal to its maximum people allowance.
5. Fill in the blanks in following query to output the IDs and names of students (other than Bob) who belonged in Fall 2023 to all of the clubs that ‘Bob’ (sid 12345) belonged to that semester:

Method 1:

```
SELECT s.sid, s.sname FROM [       ]

WHERE s.sid <> [                  ]

AND NOT EXISTS

(

-- Bob’s 2023 clubs EXCEPT this student’s clubs

   SELECT * From [    ]

   WHERE sid = 12345 AND semester = 'Fall' AND year  = 2023

   AND cid NOT in

-- Bob’s F’23 clubs

        (SELECT [                   ] FROM [                   ]

          WHERE [                   ] = s.sid AND semester = 'Fall' AND year = 2023))
```

Method 2:

```
SELECT s.sid, s.sname FROM Student s

WHERE s.sid <> 12345

AND

( -- number of clubs Bob is in during Fall 2023

SELECT COUNT(*)

  FROM Membership WHERE  [        ])

=

(

-- number of clubs Bob and current student are both in during Fall 2023

SELECT COUNT(*)
```

```
 FROM Membership AS m1 JOIN Membership AS m2 USING (cid, semester, year)
WHERE semester = 'Fall' AND year = 2023 AND [   ])
```

6. Output the sid, sname, cid, cname, for each student and club such that the student has been a member of the club every semester since Fall 2023.
7. Output the id and name of the club that either hasn't held an event or held an event that no one registered for.
8. Output the id and name of the club that has the highest average rating of events held in Fall 2023.
9. Write a TRC query corresponding to the problem 4.5 **Hint:** It should involve a construct like, for all s (P1(s) ⇒ P2(….)) where P1 and P2 are predicates involving tuples, attributes, and constants